Testing Legacy Code

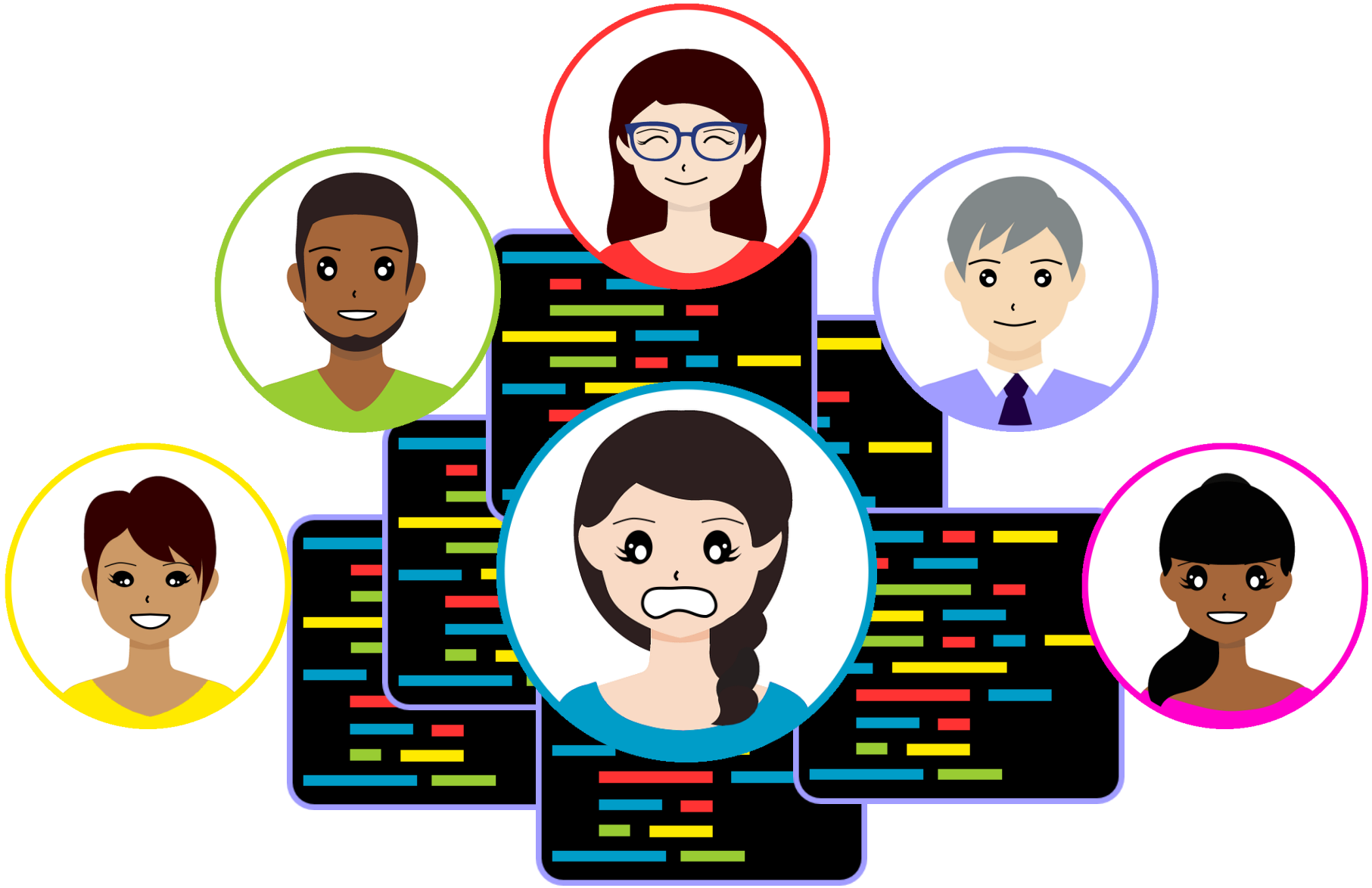
Fuzzing for Better Input Data

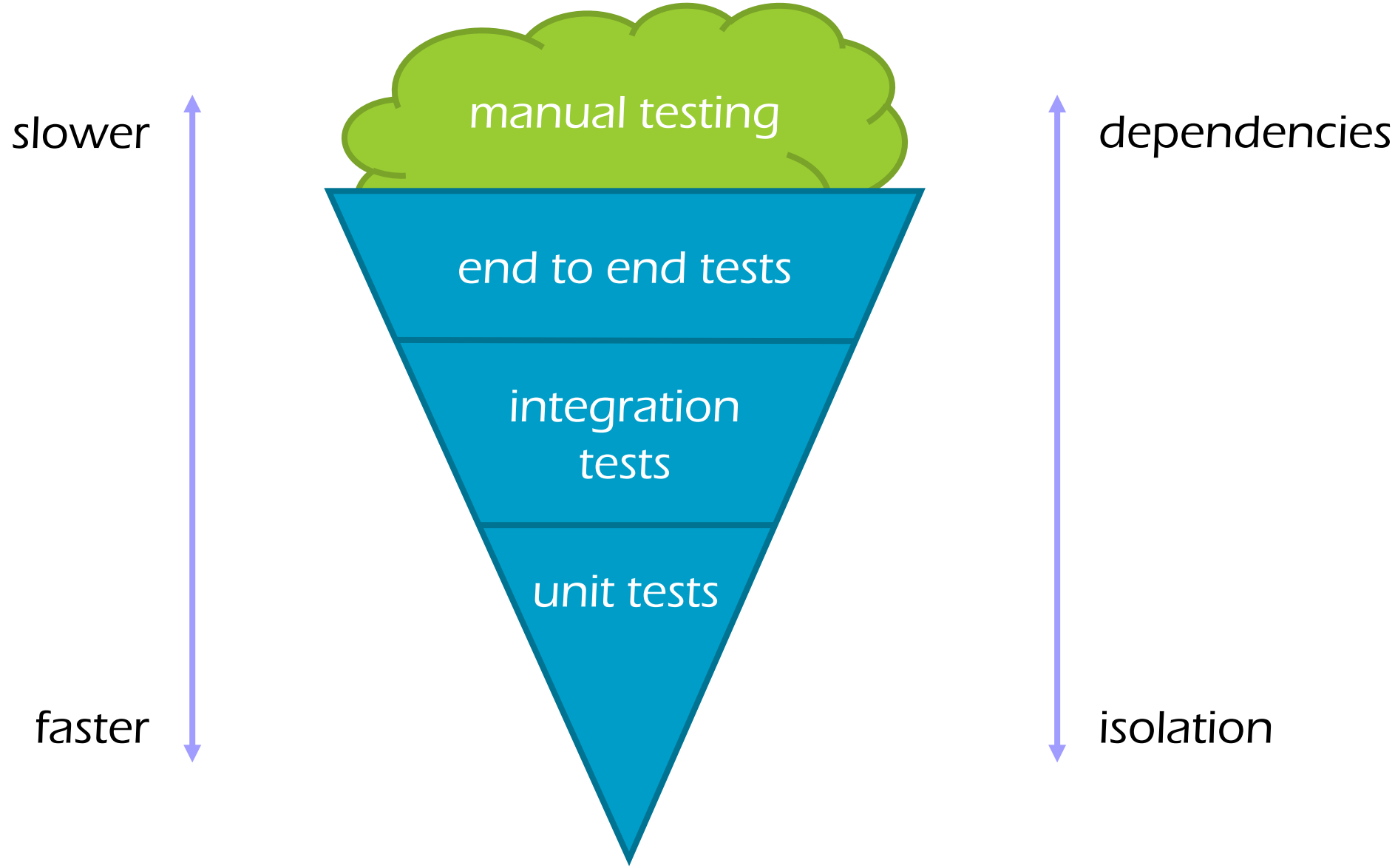
Niel Waldren
cpp_niel

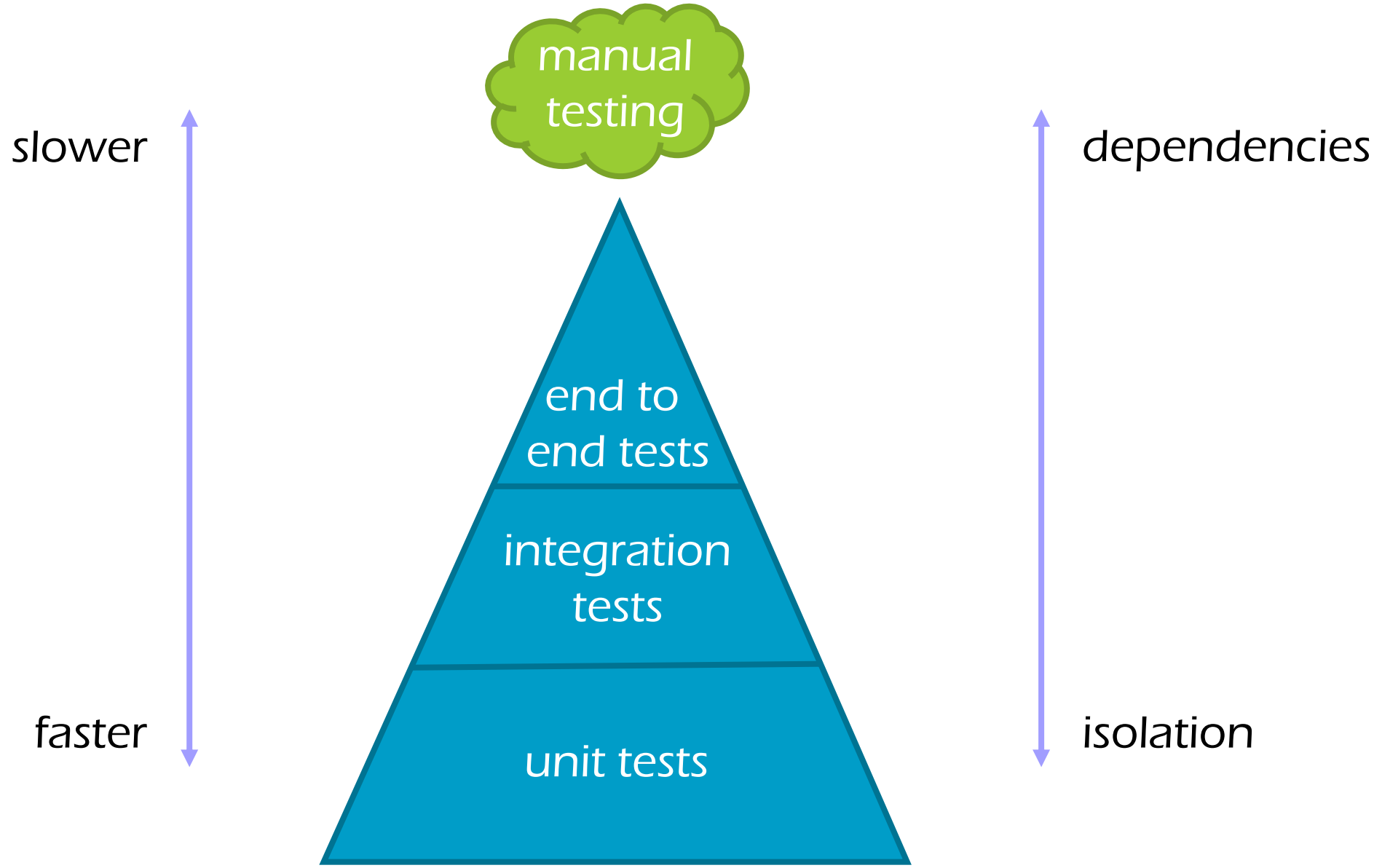
Tina Ulbrich
Yulivee

Meeting C++ 2019

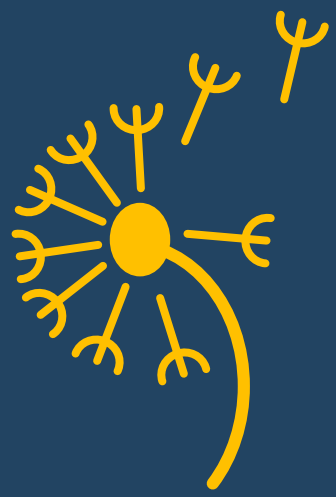
Rosen Technology and Research Center

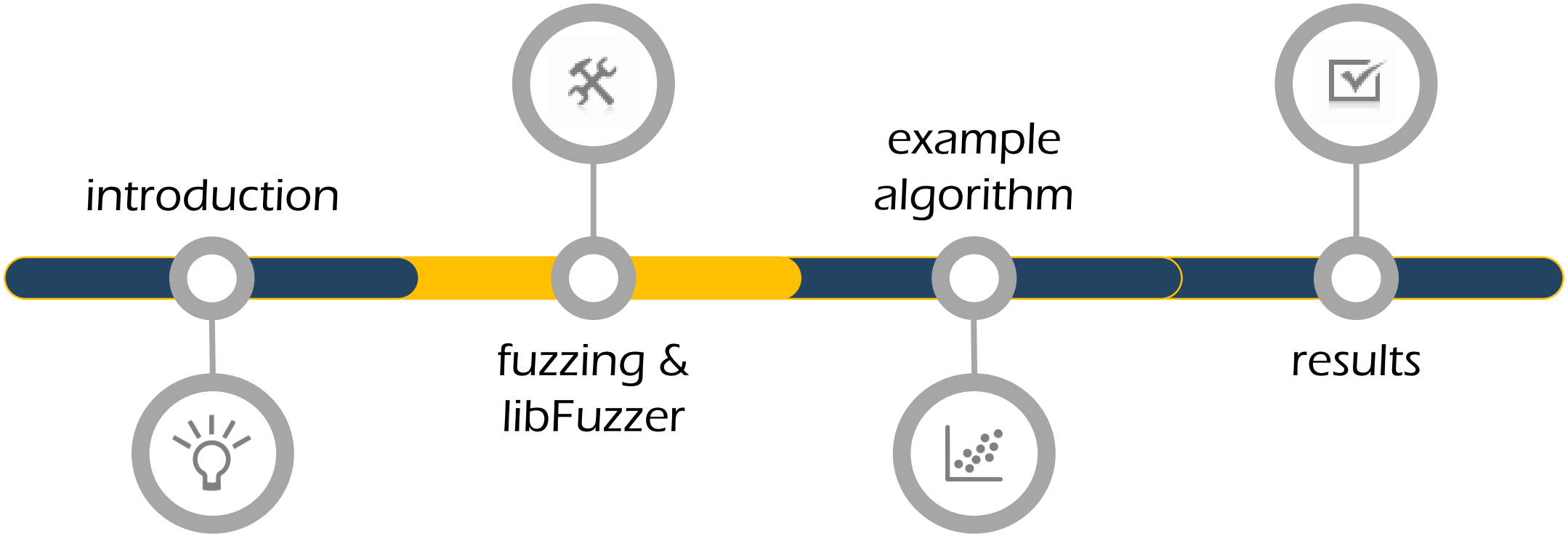


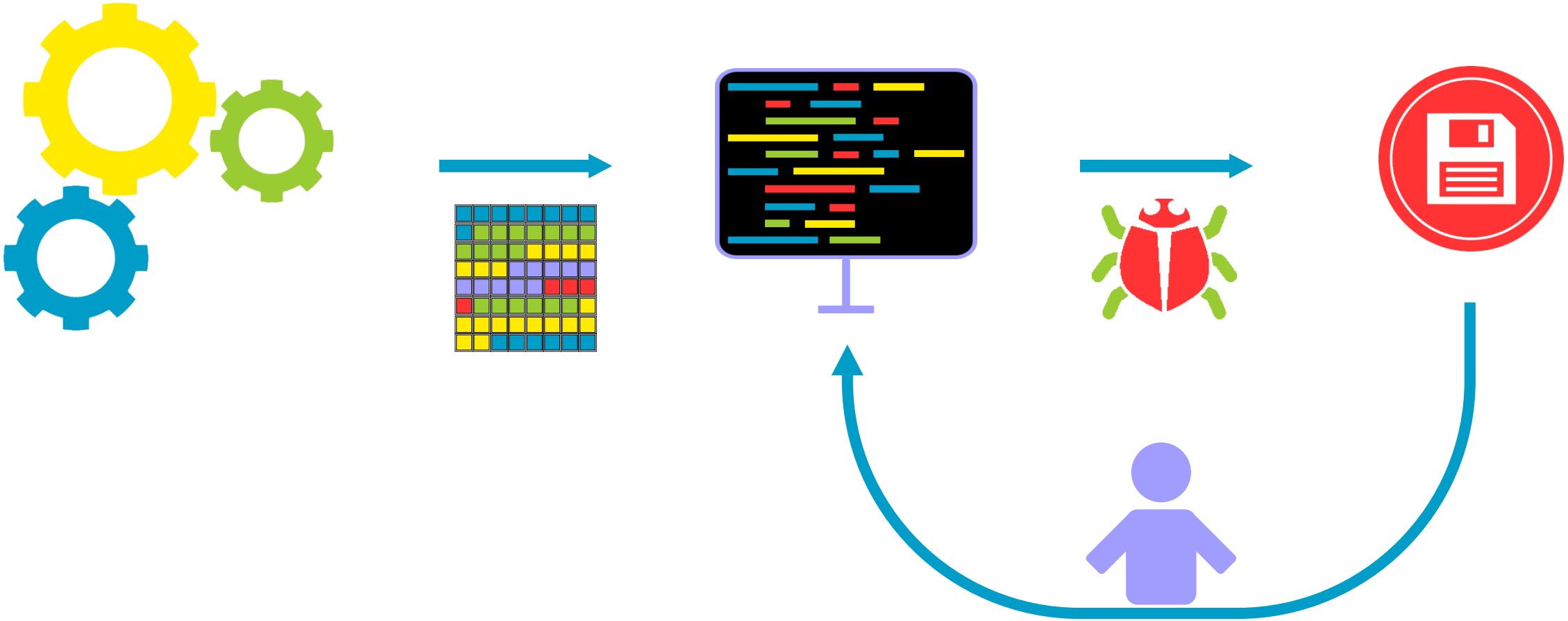

slower
faster
manual testing
end to end tests
integration tests
unit tests
dependencies
isolation

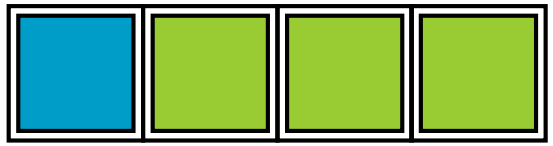

manual testing
slower
faster
dependencies
isolation
end to end tests
integration tests
unit tests

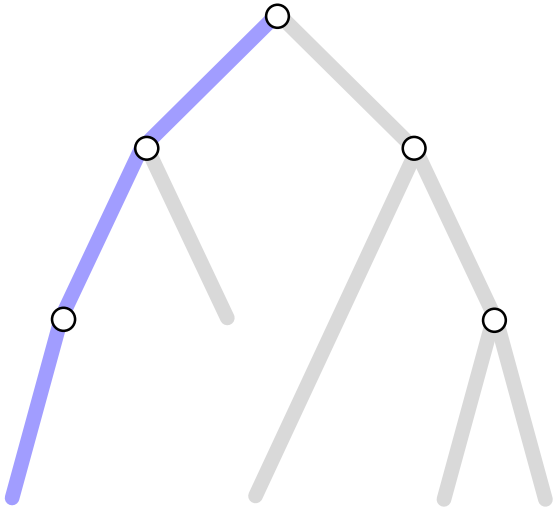
How to reduce test input data yet increase confidence in our releases?




introduction
fuzzing &
libFuzzer
example
algorithm
results



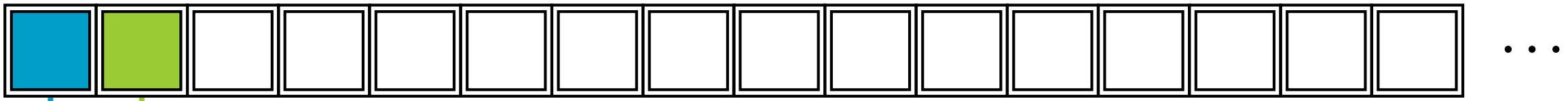




```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t* input_data, size_t size)
{
    do_something(input_data, size);
    return 0;
}
```

```
double calculate(double lower_threshold, double upper_threshold,
                 const matrix& data);


extern "C" int LLVMFuzzerTestOneInput(const uint8_t* input_data, size_t size)
{
    const auto[lower, upper, data] = create_arguments(input_data, size);
    calculate(lower, upper, data);
    return 0;
}
```

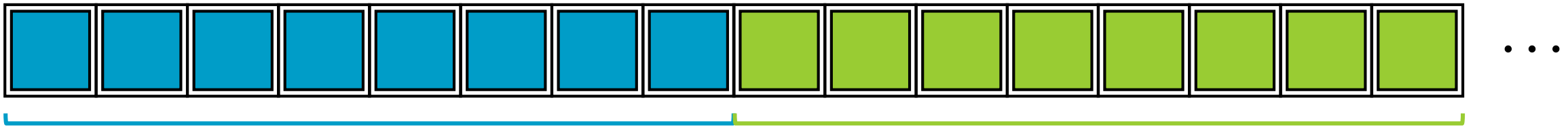
```
double calculate(double lower_threshold, double upper_threshold, ...);
```



```
const auto lower_threshold = static_cast<double>(bytes[0]);
const auto upper_threshold = static_cast<double>(bytes[1]);
```

```
if (lower_threshold > upper_threshold)
    swap(lower_threshold, upper_threshold);
```

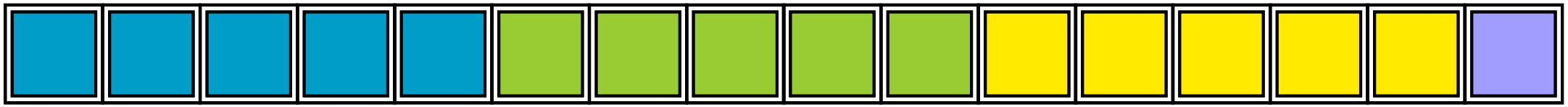
```
double calculate(double lower_threshold, double upper_threshold, ...);
```



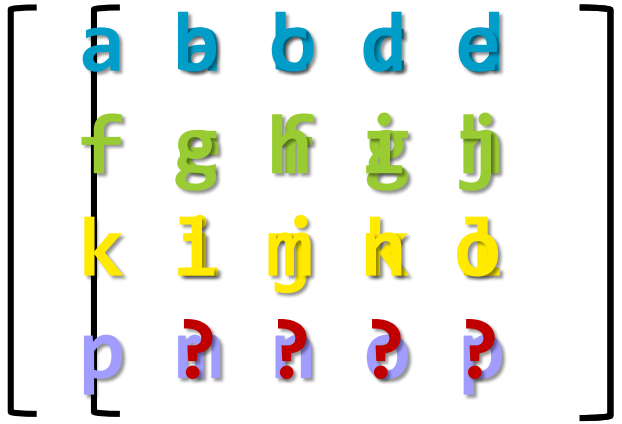
```
auto lower_threshold = 0.0;
std::copy(bytes, bytes + 8, reinterpret_cast<std::byte*>(&lower_threshold));

auto upper_threshold = 0.0;
std::copy(bytes + 8, bytes + 16, reinterpret_cast<std::byte*>(&upper_threshold));
```

```
double calculate(..., const matrix& data);
```

...

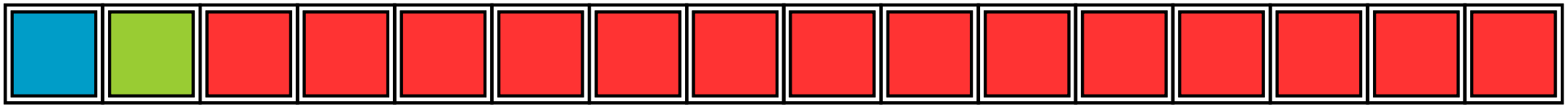


```
auto m = matrix(BYTE, 4, 4);
std::copy(bytes, bytes + 16, m.data());
```



```
double calculate(..., const matrix& data);
```

...

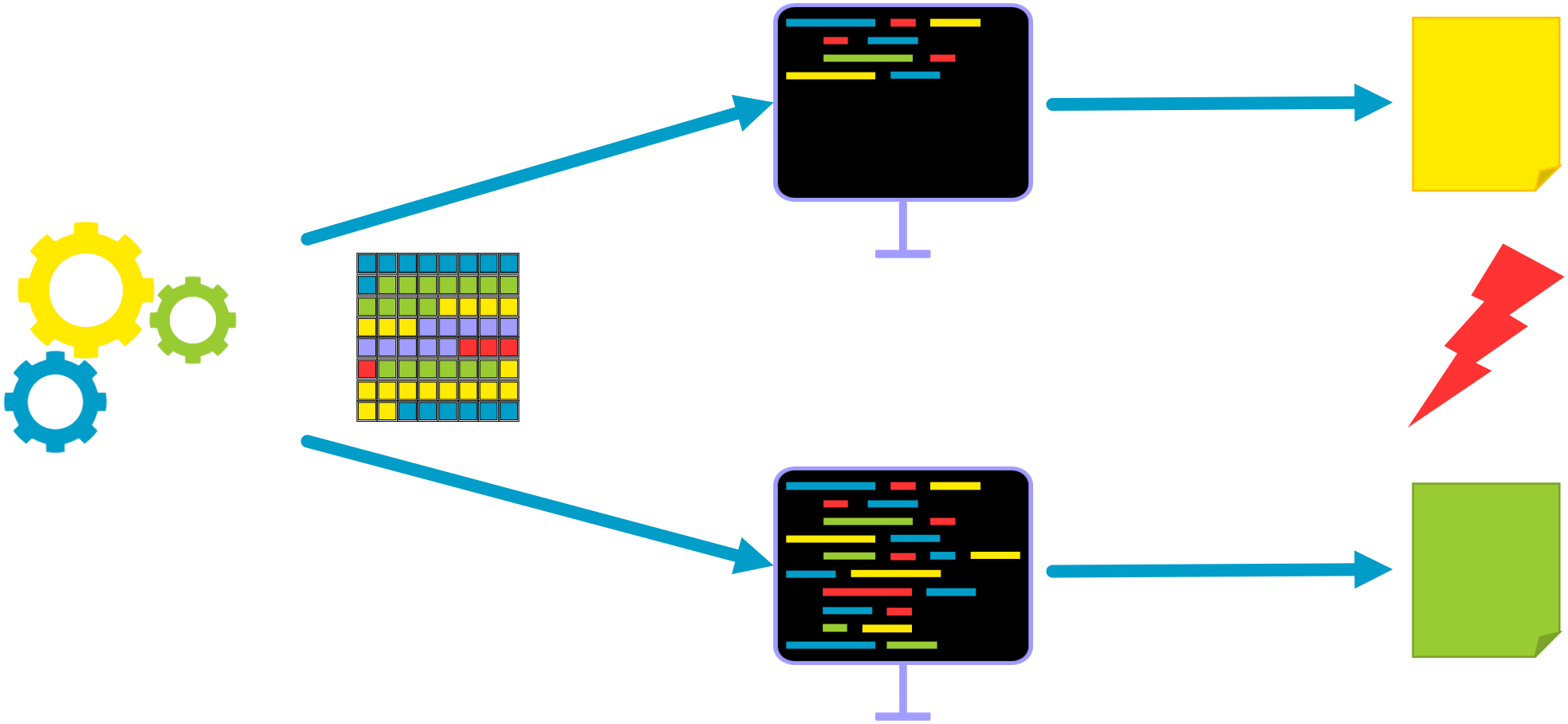


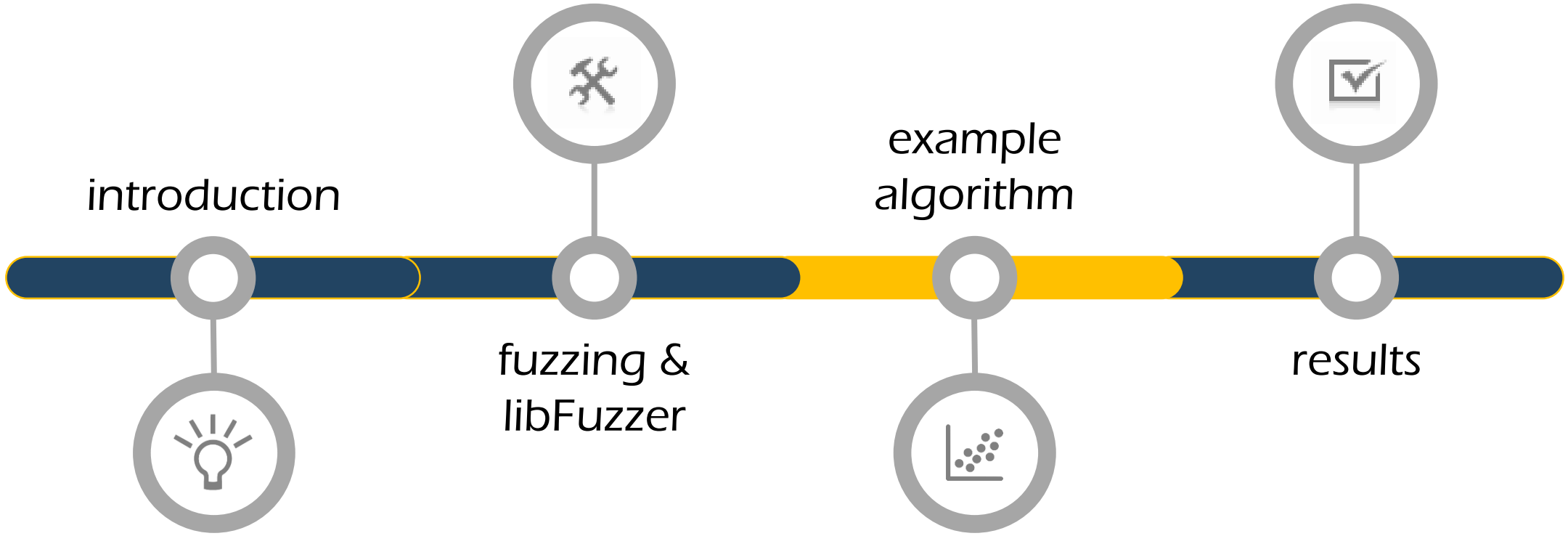
```
const auto rows = static_cast<size_t>(bytes[0]);
const auto cols = static_cast<size_t>(bytes[1]);
auto m = matrix(COMPLEX, rows, cols);
std::transform(bytes + 2, bytes + size, m.data(), ??);
```

$$\begin{bmatrix} a & \cdot & \cdot & \cdot \\ \cdot & \cdot & & \\ \cdot & & \cdot & \\ \cdot & & & \cdot \end{bmatrix}$$

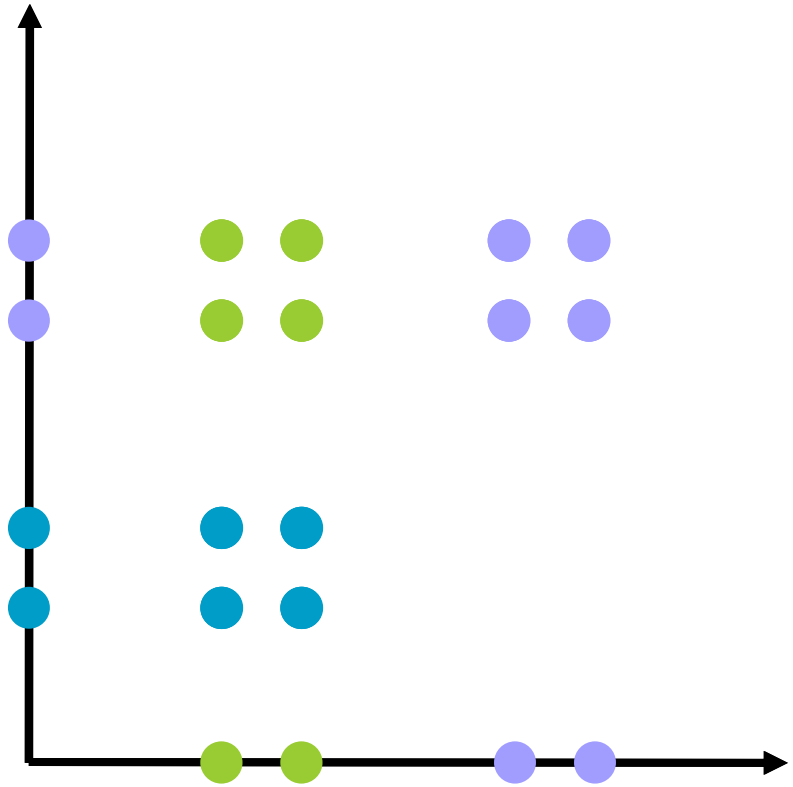
```
#128    pulse   cov: 790 ft: 1530 corp: 46/767b lim: 4 exec/s: 64 rss: 27Mb
#134    NEW     cov: 790 ft: 1532 corp: 47/792b lim: 4 exec/s: 67 rss: 27Mb L: 25/29 MS: 3 ChangeBit-ShuffleBytes-
#140    NEW     cov: 790 ft: 1533 corp: 48/800b lim: 4 exec/s: 70 rss: 27Mb L: 8/29 MS: 1 CMP- DE: "\x00\x00\x00\xc7"-
#141    NEW     cov: 790 ft: 1536 corp: 49/805b lim: 4 exec/s: 70 rss: 27Mb L: 5/29 MS: 1 EraseBytes-
#147    REDUCE  cov: 790 ft: 1536 corp: 49/801b lim: 4 exec/s: 73 rss: 27Mb L: 25/29 MS: 1 CrossOver-
#151    NEW     cov: 790 ft: 1537 corp: 50/809b lim: 4 exec/s: 75 rss: 27Mb L: 8/29 MS: 4 ChangeBinInt-EraseBytes-
#157    NEW     cov: 790 ft: 1538 corp: 51/833b lim: 4 exec/s: 78 rss: 27Mb L: 24/29 MS: 1 EraseBytes-
#173    NEW     cov: 790 ft: 1539 corp: 52/850b lim: 4 exec/s: 86 rss: 27Mb L: 17/29 MS: 1 ShuffleBytes-
```

Run Number | Event | Coverage | Corpus | Length Limit | Current Speed | Memory Use | Mutation

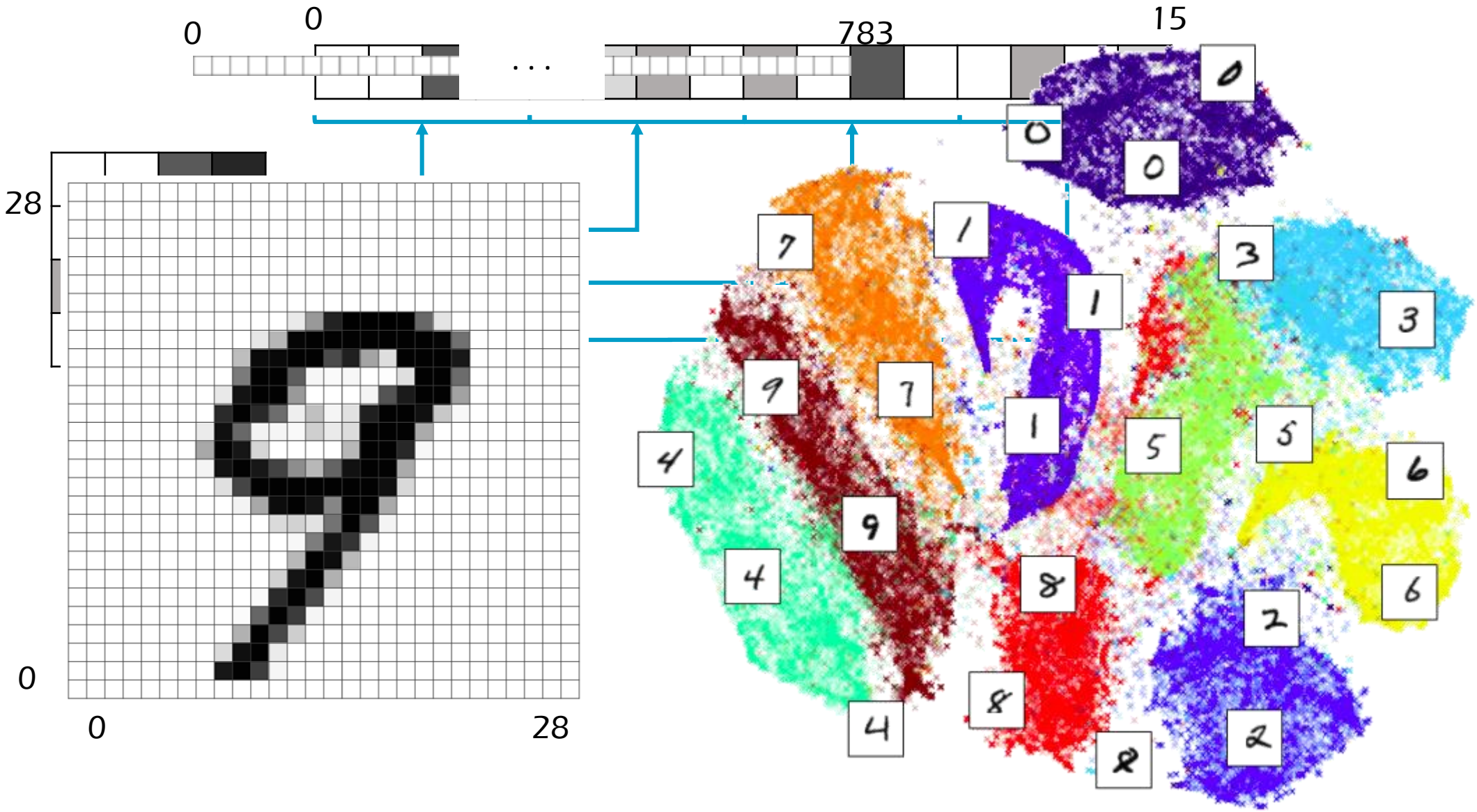



introduction
fuzzing &
libFuzzer
example
algorithm
results

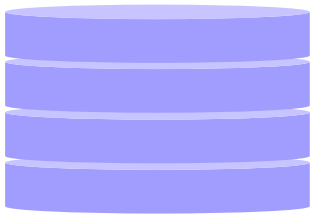
simple projection



t-SNE



https://nlml.github.io/in-raw-numpy/in-raw-numpy-t-sne/

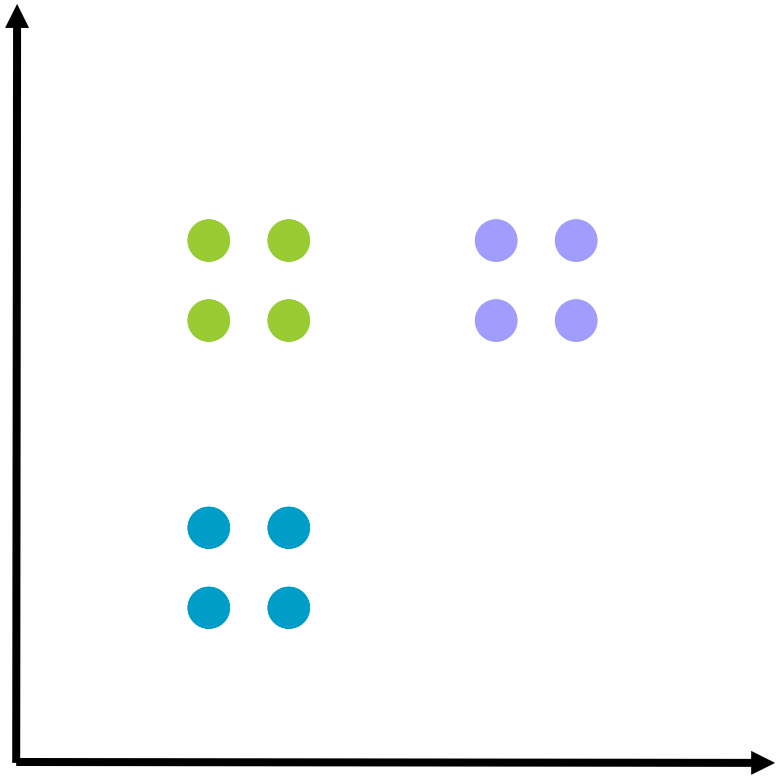


10MB

72%

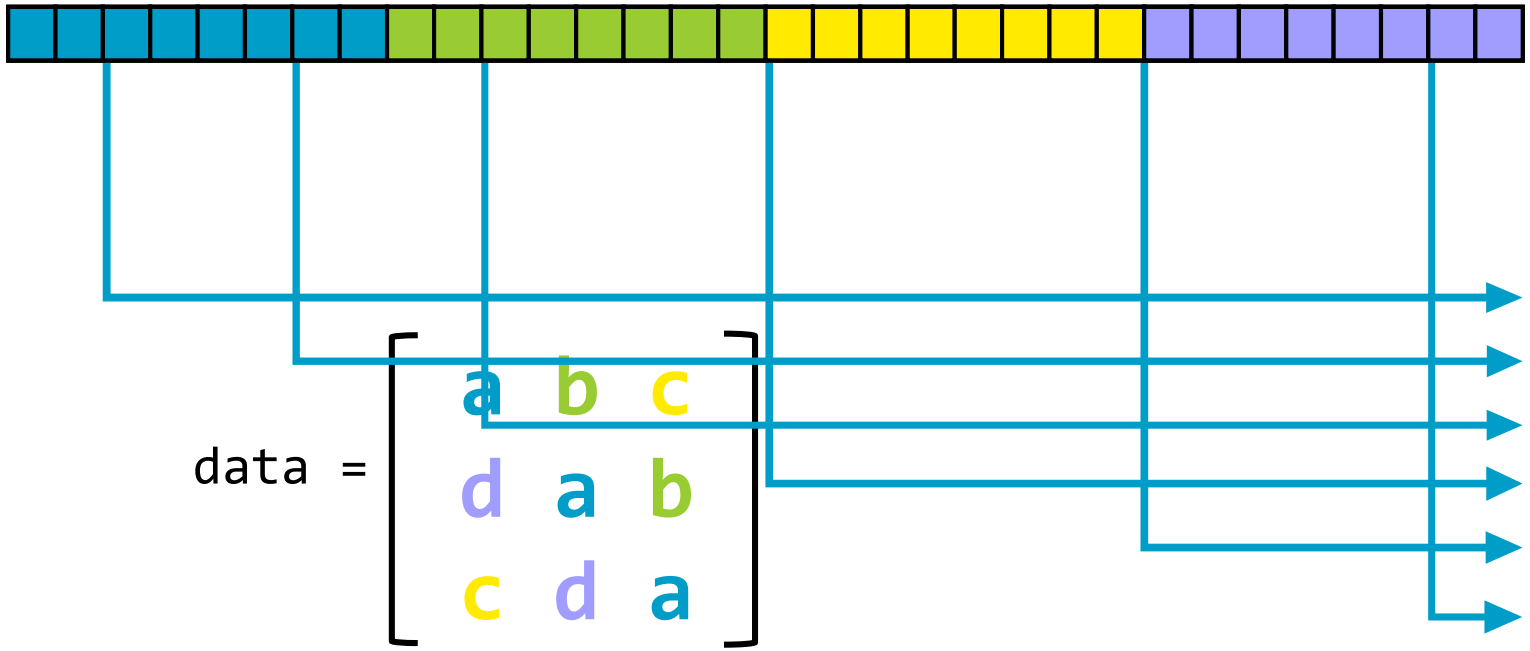
```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t* input_data, size_t size)
{

    // create arguments here

    TSNE::run(input_matrix, num_vectors, num_src_dims, output_matrix,
              num_target_dims, perplexity, theta, rand_seed, max_iter);

    return 0;
}
```

2D Source Space

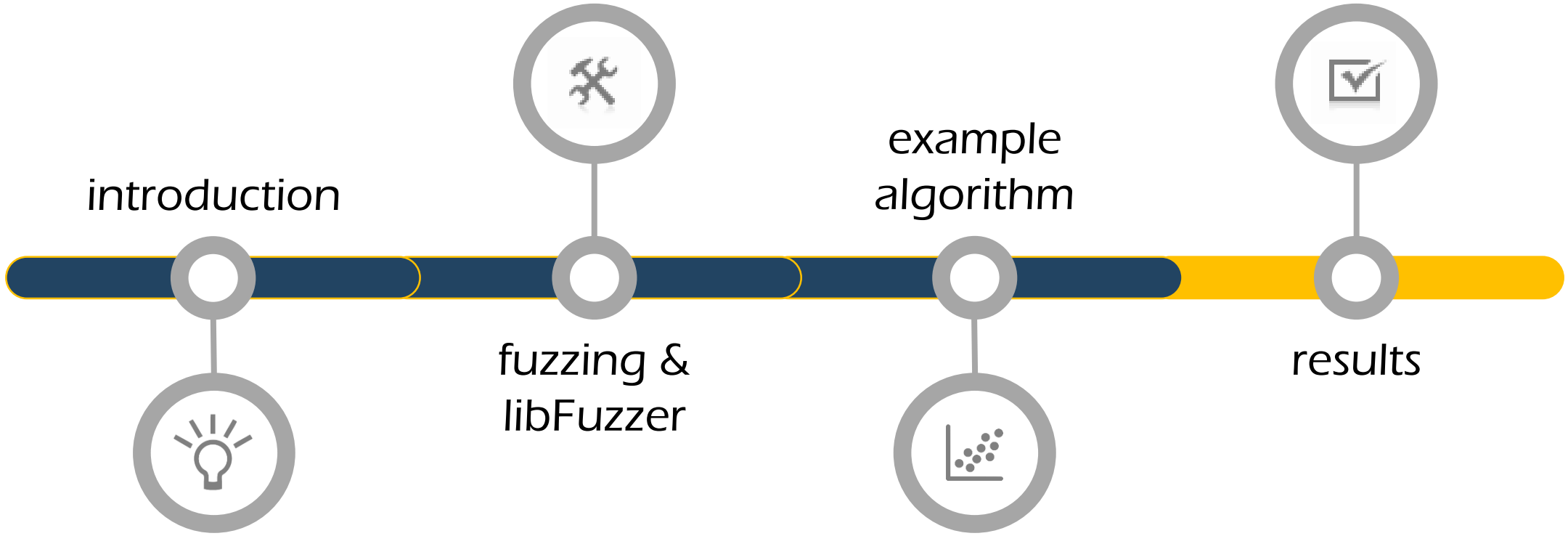


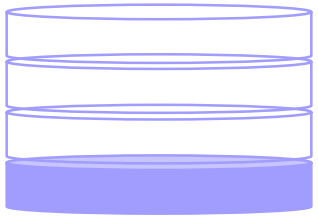
1D target space

```
num_vectors:        12
num_src_dims:        2
num_target_dims:     1
```




```
num_vectors:        3
num_src_dims:       3
num_target_dims:    1
perplexity:       NaN
theta:          123.456
seed:            42
max_iter:       312
```


introduction
fuzzing &
libFuzzer
example
algorithm
results

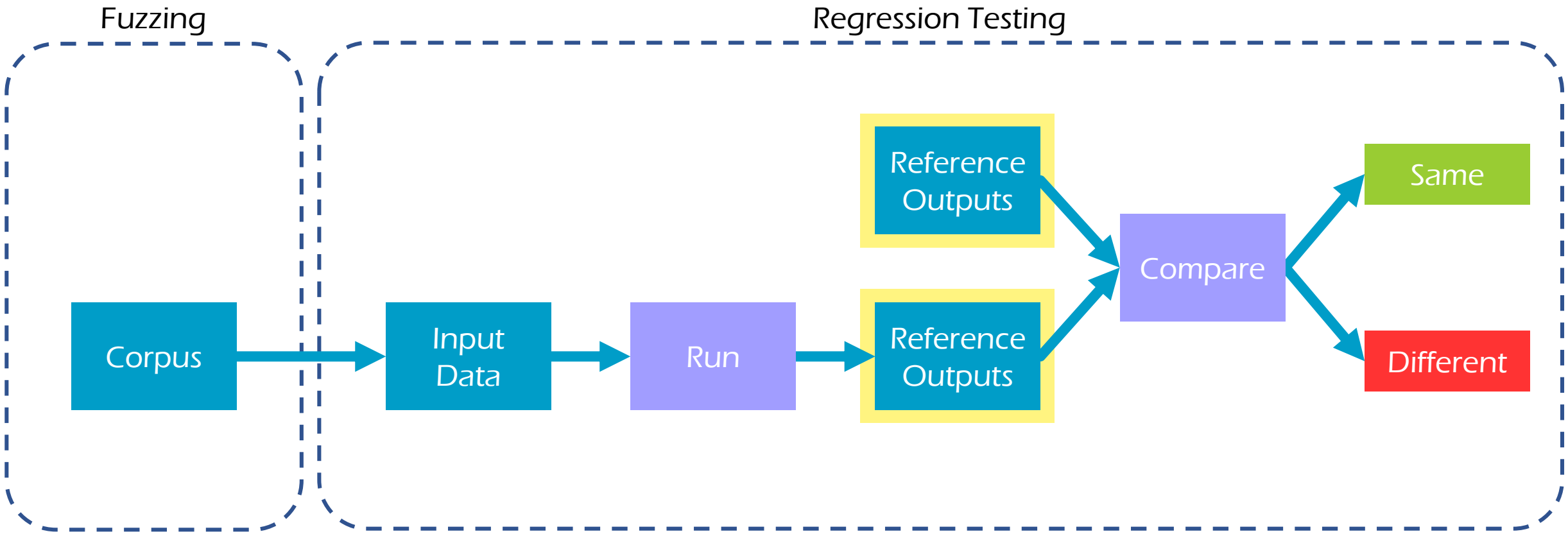


0.2KB

96%

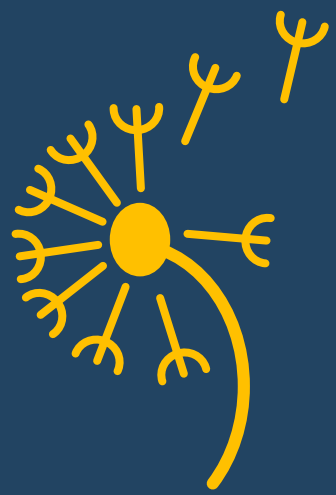
```
TEST_CASE()
{
    const auto bytes = std::array{/* bytes from a file from the corpus */};
    const auto[data, num_vectors, ...] = create_arguments(bytes);

    matrix output;
    TSNE::run(data, output, num_vectors, ...);

    const auto expected_output = matrix{...};
    EXPECT(output == expected_output);
}
```

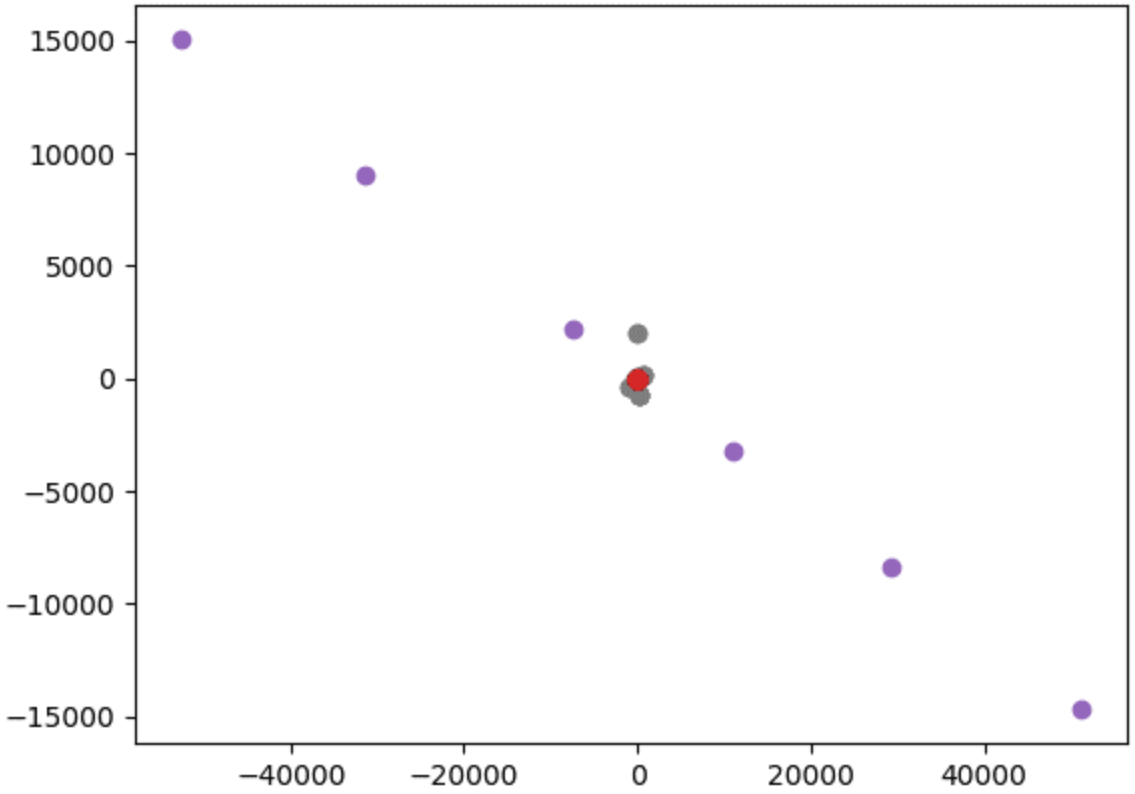
```
TEST_CASE()
{
    const auto data = matrix{...};
    const auto num_vectors = 3;
    const auto num_src_dims = 10;
    ...

    matrix output;
    TSNE::run(data, output, num_vectors, num_src_dims, ...);

    const auto expected_output = matrix{...};
    EXPECT(output == expected_output);
}
```

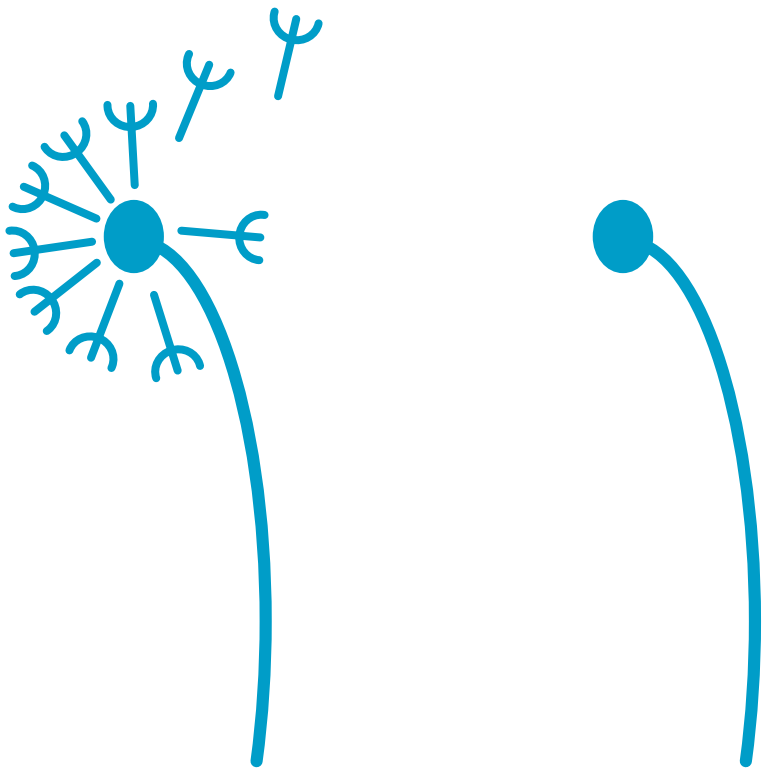

Fuzzing
Regression Testing
Corpus
Input Data
Run
Reference Outputs
Reference Outputs
Compare
Same
Different

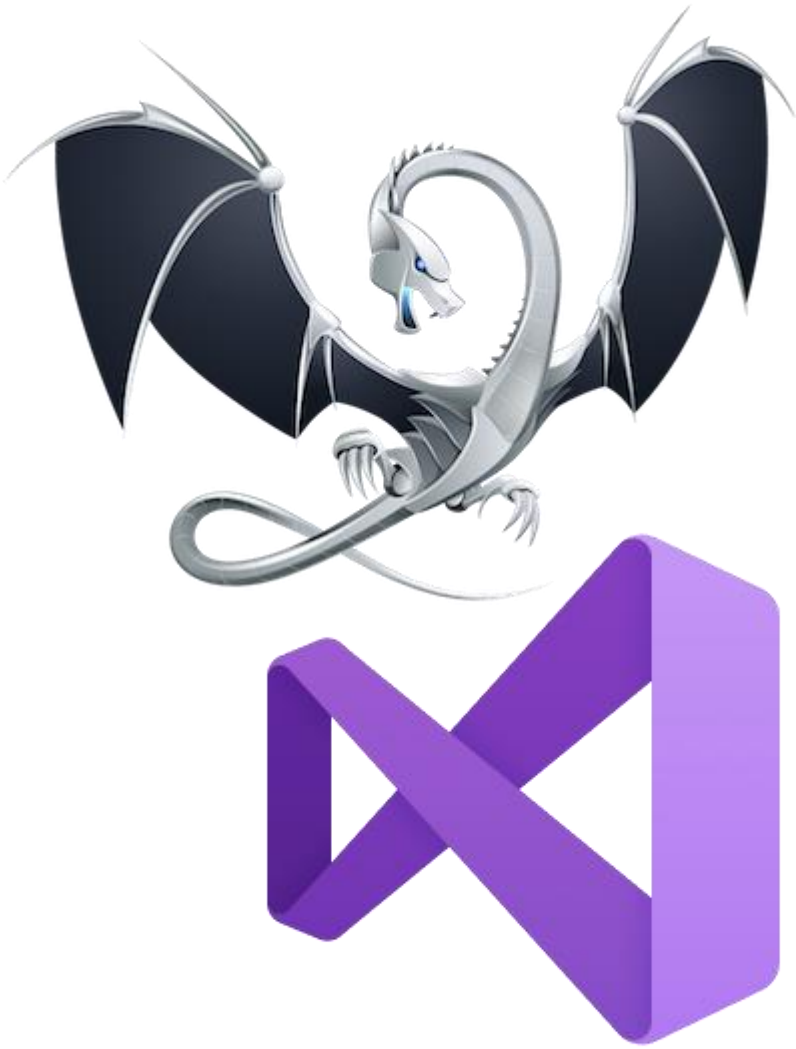
How to reduce test input data yet increase confidence in our releases?

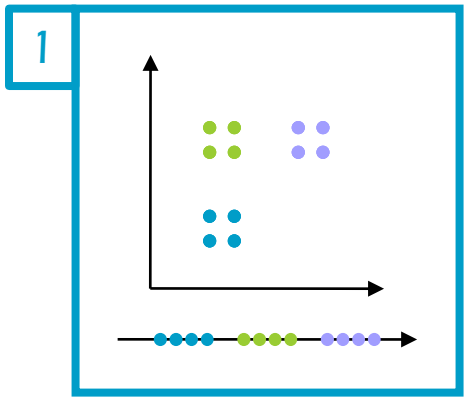
- Write the fuzz target
- Run the fuzzer
- Add test cases
- Done

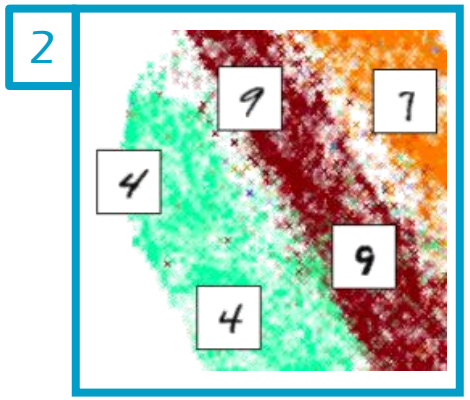


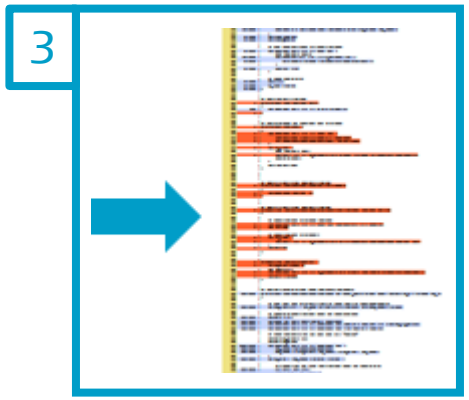

15000
10000
5000
0
−5000
−10000
−15000
−40000
−20000
0
20000
40000

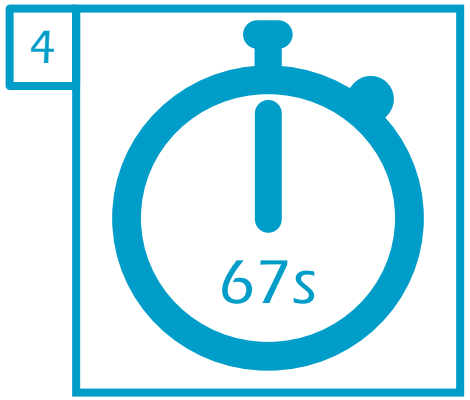


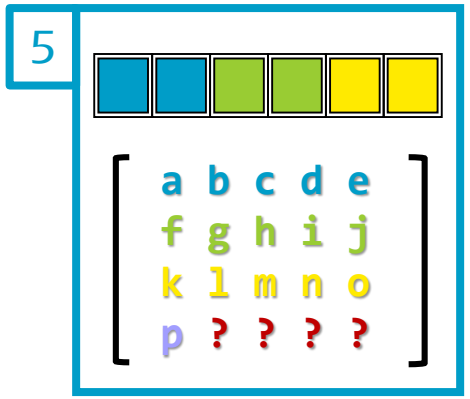


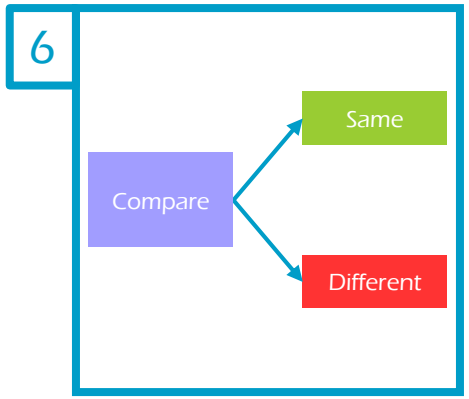

1

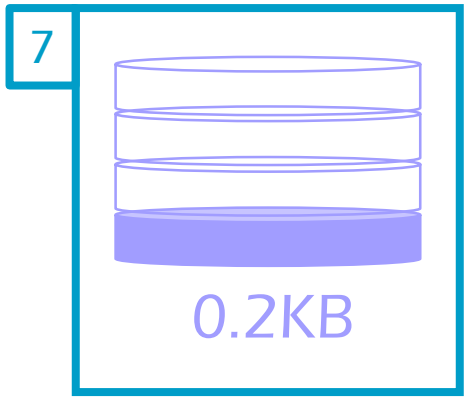

2


3

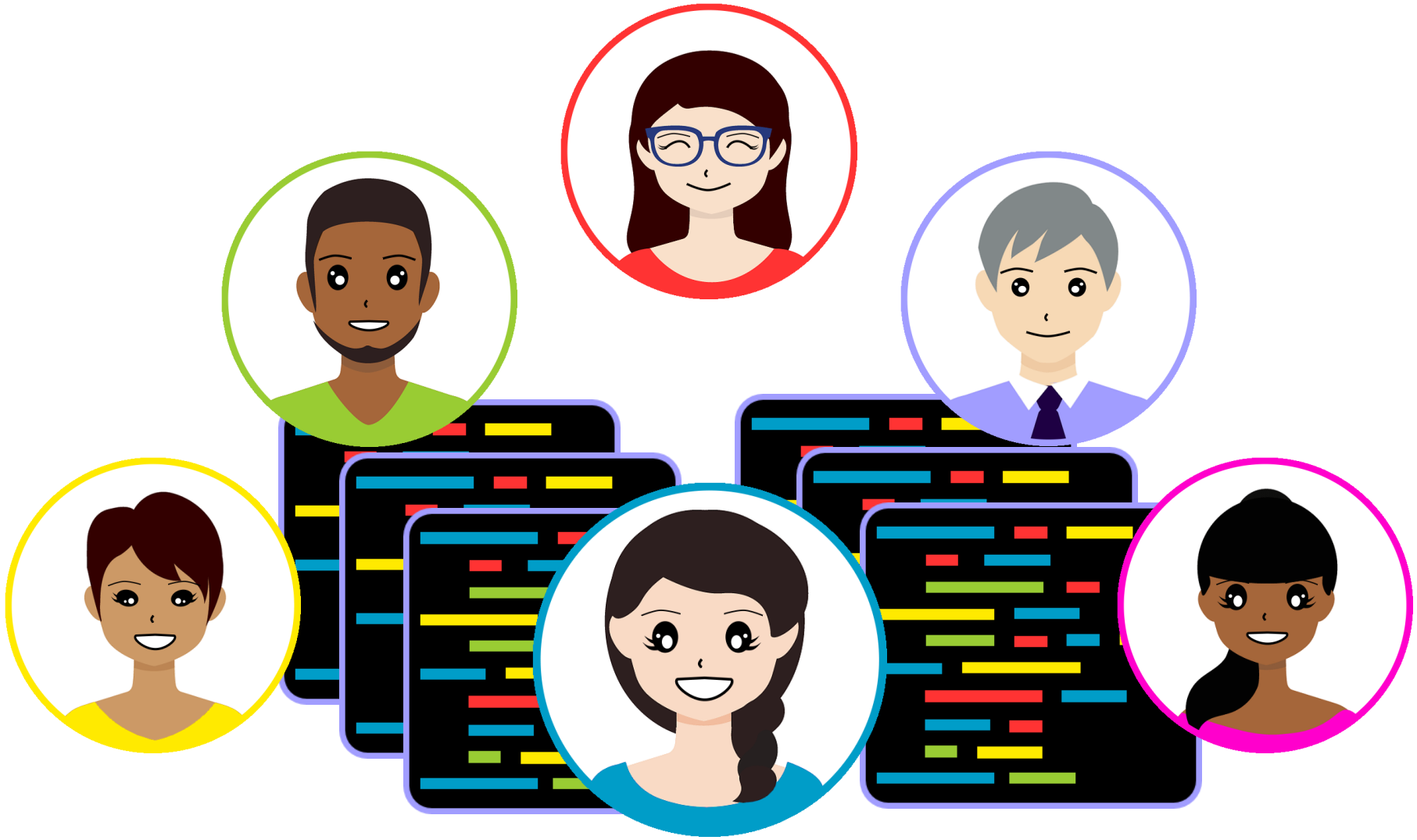

4
67s


5
a b c d e
f g h i j
k l m n o
p ? ? ? ?


6
Compare
Same
Different


7
0.2KB


8



Meeting C++ 2019

Rosen Technology and Research Center

Resources

- libFuzzer
- libFuzzer on Windows with cmake/Visual Studio
- Clang/LLVM support for Visual Studio
- AFL
- WINAFL
- t-SNE algorithm on GitHub
- an introduction to t-SNE with Python example
- In Raw Numpy: t-SNE
- Approval Testing
- Quickly Testing Legacy Code – C++ on Sea Talk
- pulling jpegs out of thin air